# NoSQL

## day01

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY01\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NoSQL

-Not Only SQL不仅仅是数据库

-泛指非关系型数据库

-不需要预先定义数据库存储结构

-每条记录可以有不同的数据类型和字段个数

NoSQL软件

Memcached Redis MongoDB CouchDB Neo4j FlockDB

RDBMS软件

MySQL MariaDB Oracle DB2 SQL Server

一: 部署redis 软件 (redis 高性能分布式内存数据库)

1. redis介绍

2. 安装redis

yum -y install gcc

tar -zxf redis-4.0.8.tar.gz //一定要先安装gcc在解压

make && makeinstall

3. 初始化配置redis

./utils/install\_server.sh //执行初始化设置脚本

Please select the redis port for this instance: [6379] //端口号

Please select the redis config file name [/etc/redis/6379.conf] //主配置文件路径及名称

Please select the redis log file name [/var/log/redis\_6379.log] //日志文件路径及名称

Please select the data directory for this instance [/var/lib/redis/6379] //数据库目录

Please select the redis executable path [/usr/local/bin/redis-server] //启动程序目录

Port : 6379 //端口号

Config file : /etc/redis/6379.conf //配置文件目录

Log file : /var/log/redis\_6379.log //日志目录

Data dir : /var/lib/redis/6379 //数据库目录

Executable : /usr/local/bin/redis-server //启动程序的目录

Cli Executable : /usr/local/bin/redis-cli //命令行的连接工具

Is this ok? Then press ENTER to go on or Ctrl-C to abort. //回车完成配置

Copied /tmp/6379.conf => /etc/init.d/redis\_6379 //服务启动脚本

4. 服务管理

/etc/init.d/redis\_6379 start //启动服务

/etc/init.d/redis\_6379 stop //停止服务

/etc/init.d/redis\_6379 status //查看服务状态

/etc/init.d/redis\_6379 restart //重新启动

netstat -antupl |grep :6379 //查看端口

5. 连接服务,存取数据

redis-cli //默认连接127.0.0.1地址的 6379端口

set key名 key值 //储存一个key值

mset key名 key值 .... //储存多个key值

get key名 //获取key值

mget key名 ... //获取多个key值

keys \* //显示所有key名

type key名 //查看key类型

exists key名 //测试key名是否存在

ttl key名 //查看key生存时间(-2已过期 -1 永久存在,单位为秒)

expire key名 数字 //设置key有效时间

move key名 库编号 //移动key到指定库

select 数据库编号0-15 //切换库

del key名 //删除指定key

flushdb //删除所在库的所有key

flushall //删除内存所有key

save //保存所有key到硬盘

shutdown //停止服务

二:

## day02

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY02\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 :集群环境准备

在51-56设置集群　6台redis服务器

　 /etc/init.d/redis\_6379 stop

　　 vim /etc/redis/6379.conf

　　 70 bind 192.168.4.51

　　　 93 port 6351

　　　 501 # requirepass foobared　　//密码

　　　 815 cluster-enabled yes　　//启用集群功能

　　　 823 cluster-config-file nodes-6379.conf　　//储备集群信息文件

　　　 829 cluster-node-timeout 5000　　//链接超时服务　　5秒

　　 rm -rf /var/lib/redis/6379/＊　　　//删除所有相关原来的东西

　　 vim +43 /etc/init.d/redis\_6379　　//修改脚本

　　 $CLIEXEC -p -h 192.168.4.51 -p 6351 shutdown

/etc/init.d/redis\_6379 start　　　　//重新启动

redis-cli -h 192.168.4.51 -p 6351

哈希池　0-16383　分3个大块　每个大块一主一从

0-5460 5461-10922 10923-16383

二 : 部署redis集群

1.配置管理主机51

部署ruby脚本运行环境

yum -y install rubygems //安装依赖包

rpm -q ruby

scp /linux-soft/03/redis/redis-3.2.1.gem root@192.168.4.51:/root/ //拷贝gem包

gem install redis-3.2.1.gem //安装

1 gem installed

创建集群管理脚本

cd redis-4.0.8/src/

mkdir /root/bin

cp redis-trib.rb /root/bin/

chmod +x redis-trib.rb

redis-trib.rb help

2. 创建集群

vim /etc/redis/6379.conf

815 cluster-enabled yes　　//启用集群功能

　　　 823 cluster-config-file nodes-6379.conf　　//储备集群信息文件

　　　 829 cluster-node-timeout 5000　　//链接超时服务　　5秒

　　　 /etc/init.d/redis\_6379 start

redis-trib.rb create --replicas 1 192.168.4.51:6351 192.168.4.52:6352 192.168.4.53:6353 192.168.4.54:6354 192.168.4.55:6355 192.168.4.56:6356

[OK] All 16384 slots covered.

redis-trib.rb create info //查看集群信息

192.168.4.53:6353> cluster info //查看集群信息

192.168.4.53:6353> cluster nodes //查看集群节点信息

redis-trib.rb check 192.168.4.52:6352 //检查集群主机角色

3. 访问集群

集群存取数据工作原理:

变量名和crc16算法做hash计算,得到一个数字 ,和16384取余,根据余数得值,找对应的redis服务器

redis-cli -c -h 192.168.4.52 -p 6352

redis-cli -c -h ip地址 -p 端口号 选项 -c 集群模式

三 : 管理redis集群

3.1 测试集群高可用功能

关闭53服务,启动53服务

redis-trib.rb check 192.168.4.56:6356

把主服务器的redis服务停止,对应的从服务器会自动升级为主服务器.

宕机的主服务器redis服务重新启动后,自动做当前主服务器的从库,并自动同步宕机期间产生的新数据

3.2 向集群中添加新主机

3.2.1 添加master角色主机 (管理主机51操作)

redis-trib.rb add-node 192.168.4.58:6358 192.168.4.53:6353 //53为master

redis-trib.rb info 192.168.4.53:6353 //查看是否添加成功

redis-trib.rb check 192.168.4.53:6353 //检测集群

redis-trib.rb reshard 192.168.4.53:6353

How many slots do you want to move (from 1 to 16384)?4096 //拿出4096个hash 槽给主机192.168.4.58

What is the receiving node ID? c5e0da48f335c46a2ec199faa99b830f537dd8a0 //主机192.168.4.58的id值

Source node #1:all //从当前所有master服务器获取hash槽

Do you want to proceed with the proposed reshard plan (yes/no)?yes //同意以上配置

redis-trib.rb info 192.168.4.53:6353

3.2.2 添加slave角色主机

redis-trib.rb add-node --slave 192.168.4.59:6359 192.168.4.51:6351 //执行添加命令

Configure node as replica of 192.168.4.58:6358. //提示添加完成

redis-trib.rb info 192.168.4.51:6351 //查看信息

redis-trib.rb check 192.168.4.53:6353 //检测集群

3.3 从集群中移除主机

3.3.1 移除slave角色主机

redis-trib.rb del-node 192.168.4.57:6357 d9fcd44a06e9661ff033c86bc1a8b071d1919c4c //执行移除命令

>>> SHUTDOWN the node. //停止移除服务的Redis服务

3.3.2 移除master角色主机

释放占用的hash槽

redis-trib.rb reshard 192.168.4.58:6358

How many slots do you want to move (from 1 to 16384)?4096 //移除4096个数槽

What is the receiving node ID? bc5c4e082a5a3391b634cf433a6486c867cfc44b

//要移动给谁的id即目标主机（这里可以随机写一个master的ID）

Source node #1: c5e0da48f335c46a2ec199faa99b830f537dd8a0 //从谁那移动即源主机,要删除主机的id

Source node #2: done //设置完毕

Do you want to proceed with the proposed reshard plan (yes/no)?yes //提交

移除主机

redis-trib.rb del-node 192.168.4.58:6358 43f844d7ff020a7650881ded2c1747a29e1bbe61

>>> Removing node 43f844d7ff020a7650881ded2c1747a29e1bbe61 from cluster 192.168.4.58:6358

>>> Sending CLUSTER FORGET messages to the cluster...

>>> SHUTDOWN the node. //关闭主机redis服务

把移除的redis主机添加到集群

1 启动redis服务(要添加的主机)

/etc/init.d/redis\_6379 start

2 重置集群信息(要添加的主机)

redis-cli -h 192.168.4.57 -p 6357

192.168.4.57:6357> cluster reset

3 添加主机到集群(管理主机)

[root@mysql51 ~]# redis-trib.rb add-node 192.168.4.57:6357 192.168.4.52:6352

4 查看集群信息

[root@mysql51 ~]# redis-trib.rb info 192.168.4.56:6356

还原操作:

停止服务

rm -rf /var/lib/redis/6379/\*

注释集群中的集群项

815 823 829

启动服务

## day03

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*NoSQL\*DAY03\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

一 : 主从复制(需要关闭集群) //主从同步加哨兵服务可以实现高可用,优点使用服务器少,两台就可以

1.1 主从复制结构模式

1.1.1一主一从

主服务器: 接收客户端访问

从服务器: 自动同步主服务器数据到本地

把主机52 配置为51的从服务器

redis-cli -h 192.168.4.51 -p 6351

192.168.4.51:6351> info replication //查看复制信息,默认都是主服务器

192.168.4.52:6352> slaveof 192.168.4.51 6351 //52设置为51的从服务器,临时有效,立即生效

[root@mysql52 ~]# vim /etc/redis/6379.conf

slaveof 192.168.4.51 6351 //永久有效

:wq

客户端50 连接51 存储数据 > set x 111

客户端50 连接52 查看数据 > keys \*

1.1.2一主多从

把主机53 也配置为51的从服务器

192.168.4.53:6353> slaveof 192.168.4.51 6351

在51主机查看复制信息 > 192.168.4.51:6351> info replication (从库个数是两个)

客户端50 连接51 存储数据 > set x 111

客户端50 分别连接从服务器 查看数据 > keys \*

1.1.3主从从

把主机54 配置为53的从服务器

192.168.4.54:6354> slaveof 192.168.4.53 6353 //把主机54 配置为53的从服务器

192.168.4.53:6353> info replication //查看复制信息

192.168.4.54:6354> info replication

客户端50 连接51 存储数据 > set x 111

客户端50 分别连接52/53/54 查看数据 > keys \*

1.1.4把从服务器恢复为独立的数据库服务器

临时恢复 > slaveof no one

192.168.4.53:6353> info replication //查看复制信息

永久恢复

[root@mysql54 ~]# vim /etc/redis/6379.conf

# slaveof 192.168.4.53 6353

:wq

[root@mysql54 ~]# /etc/init.d/redis\_6379 stop

[root@mysql54 ~]# /etc/init.d/redis\_6379 start

1.1.5配置带验证(主服务器带密码)的主从复制

给主服务器51设置连接密码

vim /etc/redis/6379.conf //修改主配置文件

501 requirepass 123456

vim +43 /init.d/redis\_6379 //修改脚本,重启服务

配置从服务器,指定主服务器的连接密码

(从服务器) vim /etc/redis/6379.conf

289 masterauth 123456

:wq

重启redis服务

info replication //查看服务状态

二 : 哨兵服务

2.1 部署一主一从服务

2.2 配置主配置文件

从服务器 viom /etc/sentinel.conf

bind 0.0.0.0 //服务地址

port 26379 //连接端口

sentinel monitor server51 192.168.4.51 6351 1

//主机名 //ip地址 //端口 //票数

sentinel auth-pass server51 123456 //连接密码

2.3 启动哨兵服务

[root@mysql52 redis-4.0.8]# redis-sentinel /etc/sentinel.conf //启动

2.4 测试

关闭主服务器

2.5 在从服务器查看复制信息(没密码的话,另一台从服务器自动做新主库从库)

info replication

2.6 查看哨兵服务的主配置文件,自动监视当前的主服务器

三 : 持久化(redis服务把内存里的数据 保存到硬盘的方式)

3.1 RDB

3.1.1 介绍

redis主配置文件 dbfilename "dump.rdb" //备份文件名

- save 300 10 //5分钟(300秒)且有10个key改变

save //阻塞写存盘

bgsave //不阻塞写存盘

3.1.2 使用RDB恢复数据

要备份的主机 :

192.168.4.50:6350> keys \*

1) "x3"

2) "x1"

192.168.4.50:6350> save

[root@mysql50 ~]# ls /var/lib/redis/6379/

dump.rdb

[root@mysql50 6379]# scp /var/lib/redis/6379/dump.rdb root@192.168.4.55:/root/

要恢复的主机 :

[root@mysql55 ~]# /etc/init.d/redis\_6379 stop

[root@mysql55 ~]# rm -rf /var/lib/redis/6379/\*

[root@mysql55 ~]# scp /root/dump.rdb /var/lib/redis/6379/

[root@mysql55 ~]# /etc/init.d/redis\_6379 start

[root@mysql55 ~]# redis-cli -h 192.168.4.55 -p 6355

192.168.4.55:6355> keys \*

1) "x"

2) "x1"

3.1.3 优缺点

优点 : 高性能的持久化实现----创建子进程来执行持久化,先写入临时文件,过程中没有IO操作

适合大规模数据恢复

缺点 : 意外宕机,丢失最后一次持久化的所有数据

3.2 AOF

3.2.1 开启AOF

192.168.4.50:6350> config set appendonly yes //开启AOF

192.168.4.50:6350> config rewrite //写入配置文件

192.168.4.50:6350> save //保存

[root@mysql50 6379]# ls /var/lib/redis/6379/ //查看

3.2.2 恢复数据

[root@mysql50 ~]# scp /var/lib/redis/6379/appendonly.aof root@192.168.4.55:/root/ //拷贝备份文件

[root@mysql55 ~]# /etc/init.d/redis\_6379 stop //停服务

[root@mysql55 ~]# vim /etc/redis/6379.conf //改配置文件

673 appendonly yes

[root@mysql55 ~]# rm -rf /var/lib/redis/6379/\* /删除数据库目录下的信息

[root@mysql55 ~]# cp /root/appendonly.aof /var/lib/redis/6379/ //拷贝备份文件

[root@mysql55 ~]# /etc/init.d/redis\_6379 start //启动

[root@mysql55 ~]# redis-cli -h 192.168.4.55 -p 6355

192.168.4.55:6355> keys \* //查看数据

3.2.3 优化配置

702 appendfsync always //实时记录,磁盘同步

703 appendfsync everysec //每秒记录,磁盘同步

704 appendfsync no //写入aof,不磁盘同步

744 auto-aof-rewrite-percentage 100 //再次重写触发值

745 auto-aof-rewrite-min-size 64mb //首次重写触发值

3.2.4 修复AOF文件

[root@mysql50 6379]# redis-check-aof --fix appendonly.aof

0x 109: Expected prefix '\*', got: 's'

AOF analyzed: size=276, ok\_up\_to=265, diff=11

This will shrink the AOF from 276 bytes, with 11 bytes, to 265 bytes

Continue? [y/N]: y

Successfully truncated AOF

[root@mysql50 6379]# rm -rf /var/run/redis\_6379.pid

[root@mysql50 6379]# /etc/init.d/redis\_6379 start

[root@mysql50 6379]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

3.2.5 优缺点

优点 : 灵活设置持久化方式

出现意外宕机时,尽可能丢失一秒数据

缺点 : 体积大于RDB方式

执行fsync策略时速度可能会比RDB方式慢

四 数据类型

4.1 string字符串

1) set key value [ex seconds] [px milliseconds] [nx | xx]

设置key及值 过期时间秒 毫秒 不存在就改|存在就改

2) setrange key offset values

--从偏移量开始复写key的特定位的值(修改指定字段)

192.168.4.50:6350> set aaa 123abc456

OK

192.168.4.50:6350> SETRANGE aaa 3 "nnn"

(integer) 9

192.168.4.50:6350> get aaa

" 123nnn456"

3) strlen key 统计字串长度

192.168.4.50:6350> strlen aaa

(integer) 9

4) append key value

存在则追加,不存在则创建key及value,返回key长度

192.168.4.50:6350> get x1

"1130"

192.168.4.50:6350> append x1 30

(integer) 6

192.168.4.50:6350> get x1

"113030"

5) setbit key offset value

对key所储存的子串,设置或清除特定偏移量(特定位置)上的位(bit)

value的值可以为1或0,offset为0~2^32之间

key不存在,则创建

192.168.4.51:6379> setbit bit 0 1 //设置bit第0位为1

192.168.4.51:6379> setbit bit 1 0 //设置bit第1位为0

6) bitcount key

统计子串中被设置为1的比特位数量

192.168.4.51:6379> setbit bits 0 1 //0001

192.168.4.51:6379> setbit bits 3 1 //1001

192.168.4.51:6379> bitcount bits //结果为2

192.168.4.51:6379> setbit peter 100 1 //网站上线100天用户登录了一次

192.168.4.51:6379> setbit peter 105 1 //网站上线105天用户登录了一次

192.168.4.51:6379> bitcount peter

7) decr key

将key中的值减1，key不存在则先初始化为0，再减1

192.168.4.51:6379> decr bb

192.168.4.51:6379> decr bb

8) decrby key decrement

将key中的值，减去decrement

192.168.4.51:6379> set count 100

192.168.4.51:6379> DECRBY cc 20 //定义每次减少20（步长）

192.168.4.51:6379> DECRBY cc 20

9) getrange key start end

返回字串值中的子字串，截取范围为start和end，负数偏移量表示从末尾开始计数，-1表示最后一个字符，-2表示倒数第二个字符

192.168.4.51:6379> set x 123456789

192.168.4.51:6379> getrange x -5 -1

"56789"

192.168.4.51:6379> getrange x 0 4

"12345"

10) incr key

将key的值加1，如果key不存在，则初始为0后再加1，主要应用为计数器

192.168.4.51:6379> set page 20

192.168.4.51:6379> incr page

(integer) 21

11) incrby key increment

将key的值增加increment

192.168.4.51:6379> set x 10

192.168.4.51:6379> incr x

(integer) 11

192.168.4.51:6379> incr x

(integer) 12

12) incrbyfloat key increment

为key中所储存的值加上浮点数增量 increment

192.168.4.51:6379> set num 16.1

192.168.4.51:6379> incrbyfloat num 1.1

"17.2"

4.2 list 列表

4.2.1简介:

一个变量存储多个值

先进后出

Redis的list是一个字符队列

4.2.2 list操作

1) lpush key value [ value ]

将一个或多个值value插入到列表key的表头

key不存在,则创建

lpush list a b c d

2) lrange key start stop

从开始位置读取keyd的值到stop结束

lrange list 0 2

lrange list 0 -1

lrange list 0 -2

3) lpop key

移除并返回列表头元素数据，key不存在则返回nil

192.168.4.51:6379> lpop list //删除表头元素，可以多次执行

4) llen key

返回列表key的长度

192.168.4.51:6379> llen list

5) lindex key index

返回列表中第index个值

192.168.4.51:6379> lindex list 1

6) lset key index value

将key中index位置的值修改为value

192.168.4.51:6379> lpush list a b c d

(integer) 5

192.168.4.51:6379> lset list 3 test //将list中第3个值修改为test

OK

7) rpush key value [value…]

将value插入到key的末尾

192.168.4.51:6379> rpush list3 a b c //list3值为a b c

(integer) 3

192.168.4.51:6379> rpush list3 d //末尾插入d

(integer) 4

8) rpop key

删除并返回key末尾的值

192.168.4.51:6379> RPOP list3

"d"

4.3 hash表

4.3.1简介: 一个变量有多个列,每个列有不同值

4.3.2 Hash表操作

1) hset key field value

将hash表中field值设置为value

192.168.4.51:6379> hset site google 'www.g.cn'

(integer) 1

192.168.4.51:6379> hset site baidu 'www.baidu.com'

(integer) 1

2) hget key filed

获取hash表中field的值

192.168.4.51:6379> hget site google

"www.g.cn"

3) hmset key field value [field value…]

同时给hash表中的多个field赋值

192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com

4) hmget key field [field…]

返回hash表中多个field的值

192.168.4.51:6379> hmget site google baidu

1) "www.g.cn"

2) "www.baidu.com"

5) hkeys key

返回hash表中所有field名称

192.168.4.51:6379> hmset site google www.g.cn baidu www.baidu.com

192.168.4.51:6379> hkeys site

1) "google"

2) "baidu"

6) hgetall key

返回hash表中所有key名和对应的值列表

192.168.4.51:6379> hgetall site

1) "google"

2) "www.g.cn"

3) "baidu"

4) "www.baidu.com"

7) hvals key

返回hash表中所有key的值

192.168.4.51:6379> hvals site

1) "www.g.cn"

2) "www.baidu.com"

8) hdel key field [field…]

删除hash表中多个field的值，不存在则忽略

192.168.4.51:6379> hdel site google baidu

(integer) 2